Intel® Authoring Tools for UPnP* Technologies

Research & Development at Intel

(Version 1.00, 05-07-2003)

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining applications. Intel may make changes to specifications and product descriptions at any time, without notice.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* UPnP is a certification mark of the UPnP Implementers Corp. Other names and brands may be claimed as the property of others.

Copyright © 2003 Intel Corporation

Introduction

Intel® is committed to making seamless interoperability of CE (Consumer Electronic) and PC devices in the Digital Home a reality. Intel is equally committed to UPnP* technology over IP (Internet Protocol) as the foundation of a network whose digital endpoints communicate using a set of common and interoperable protocols.

For this new digital ecosystem to function, vendors must provide consumers with software and devices enabled with UPnP technology. When vendors incorporate UPnP architecture into existing products, they add to their value by enabling them to interact with other UPnP devices.

Vendors will face the usual challenges as they upgrade existing products with UPnP capabilities. To help vendors accomplish this in the shortest possible time and at the lowest possible cost, Intel is providing Intel Authoring Tools for UPnP Technologies, a package that includes a set of tools, UPnP stacks, and source code. With this package, developers with only a minimal knowledge of UPnP architecture will be equipped to quickly enable existing products with UPnP technology.

The Challenges

Understanding

The first challenge for a developer is understanding UPnP technology. The UPnP framework is relatively new, and many engineers are not familiar with it. Reading a UPnP specification can be an intimidating task for the uninitiated. Like any new technology, acquiring a thorough understanding requires training, and training costs time and money. These are scarce commodities in today's highly competitive environment of low budgets and tight deadlines. The Intel Authoring Tools for UPnP Technologies automate and prepackage most UPnP implementation tasks, dramatically reducing the learning curve and level of understanding required. Engineers with only a minimal knowledge of UPnP technology can now implement UPnP solutions within hours of acquiring this software.

Interoperability

The next challenge is interoperability. Even when a developer understands the UPnP specifications, there is no guarantee that the UPnP solution will be interoperable with other UPnP devices. Even the best implementations and specifications are subject to field testing and practical interoperability requirements. Acquiring the skills needed to build interoperable UPnP stacks can take years and any number of airline tickets to various UPnP plugfest locations. Because interoperability is the highest priority for Intel UPnP solutions, Intel continually tests its UPnP stacks with other implementations.

Code and Run-Time Size

The third challenge is the code and run-time size of a UPnP stack. This is an important consideration when building small, embedded devices with limited memory. In order to keep costs low, a UPnP stack must be as small as possible, and run using the least amount of runtime memory. In contrast to what developers are used to, UPnP stacks generated with Intel Authoring Tools for UPnP Technologies should have a code size of only 50 to 100k for a complete UPnP solution.

Execution Performance

The last challenge relates to execution performance. Even though many devices do not have powerful, high-end processors, they still have the need to

provide an excellent user experience that includes prompt responses to UPnP requests. In many cases, the device's processor handles additional non-UPnP work, which can lead to usability problems if the UPnP stack is not designed appropriately. Because Intel is committed to providing high performance software stacks, all Intel UPnP solutions have been designed and tested on embedded device targets. The results demonstrate that Intel software components are more than capable of handling multiple, simultaneous requests.

Authoring a UPnP Solution

Building UPnP technology into a networked device or software application involves three basic steps:

1. Designing the device or control point interfaces
2. Implementing a UPnP stack that matches these interfaces
3. Validating the resulting stack for compliance and interoperation

In the Fall of 2002, Intel introduced the precursor to Intel Authoring Tools for UPnP Technologies, a suite of tools known as Intel Tools for UPnP Technologies. This package features ten tools, many of which are useful in the last step of implementing a UPnP solution—validation. It also includes Intel Service Author, which is an effective tool for the first step—designing the UPnP interfaces. Since its introduction, Intel Tools for UPnP Technologies has been available for download on the Intel website at: http://www.intel.com/labs/connectivity/upnp/

More recently, the Intel Authoring Tools for UPnP Technologies package was added to the Intel website. These authoring tools are the missing components that enable a developer to complete the first and second steps. They provide the means to create UPnP stacks that target a vendor's requirements. In addition, the Intel Authoring Tools contain UPnP solution samples that vendors can integrate into their products.

The combination of the Intel Tools and Intel Authoring Tools creates a complete solution for quickly designing, building, and testing UPnP devices and control points. Intel recommends that developers become familiar with the Intel Tools for UPnP Technologies before using the Intel Authoring Tools for UPnP Technologies.

The Generic Stack Problem

Many UPnP stacks in the industry are generic stacks, and, like TCP/IP stacks, they are a one-size-fits-all solution. Many vendors of UPnP devices, desiring the potential for code reuse and reduced maintenance, built their UPnP architecture as a generic component that could be used by any number of applications.

However, there are problems with this approach. To provide the necessary flexibility, generic UPnP stacks must have a certain amount of overhead. This makes them larger and more difficult for developers to use. Furthermore, the generic nature of the stack requires a generic interface. In addition, developers often have to implement an abstraction layer above the generic stack, or implement their application layer directly on a generic interface. This latter method may require the developer to understand the terminology of a UPnP service. Regardless of their choice, developers are likely to loose time learning about any number of UPnP details.

The Intel Approach—Code Generation

Intel addresses the challenges of UPnP implementation with a proven approach—code generation. Used in the past to solve many problems

in computer science, code generation is an application that writes source code based on a set of requirements much like a human engineer would. Code generation applications are sometimes called code wizards; Microsoft* Visual Studio* has a code wizard. With a code generation tool, a developer can quickly enter the product's requirements and leverage years of knowledge from other developers.

Intel Device Builder

Intel® Device Builder automatically aggregates well-formed service control protocol documents (SCPDs) and generates embedded code and a sample application. This frees developers from the tedious, time consuming task of creating UPnP stack-level logic, and allows them to focus instead on development of device-level logic.

Broad Application

Intel Device Builder applies to both UPnP devices and UPnP control points, providing building blocks for both. It also allows for the creation of UPnP devices containing embedded UPnP devices. With its ability to generate source code for Win32*, Posix*, and PocketPC*[1]. Intel Device Builder also offers platform flexibility.

Easy to Use

Intel Device Builder has many benefits, including ease-of-use. In addition to generating the source code for a UPnP stack, it also generates a complete sample application that includes "Place code here" comments. These comments, as well as identifying areas that need work, intuitively describe the interface into the UPnP stack. Since the UPnP stack generated by Intel Device Builder checks and converts all incoming UPnP arguments into native data types, developers can proceed with implementation without the worry of understanding underlying UPnP data types or invalid UPnP arguments.

Trouble-Free Process

Traditional UPnP development methods have been problematic because they often rely on the manual creation and editing of UPnP XML documents and the manual tracking of dependencies between XML documents. Intel Device Builder automates these tasks, eliminating human error.

Reduced Time-to-Market

After using Intel Service Author to build well-formed UPnP service descriptions, the developer uses Intel Device Builder to generate source code for a UPnP stack that exactly fits the product specifications. In a matter of minutes a process that previously took weeks or months is completed.

Small, Fast UPnP Stacks

Another benefit of Intel Device Builder code stacks is their small size. Depending on the type of device or control point, most generated UPnP stacks compile to between 50 and 100k[2]. Since many

[1] There have been discussions about increasing the target platforms to include other embedded operating systems.

[2] When computing code size, many vendors do not include the device and service XML documents; with Intel Device Builder, these documents are included within the code and comprise about 20% of total code size on a device stack. Some vendors even choose to exclude the HTTP web server modules required by UPnP; Intel includes these numbers in all code size claims.

decisions can be resolved at code generation time with Intel Device Builder, the resulting code is smaller and more efficient. For example, outbound XML can be pre-generated, and string parsing can be used to parse XML with expected tags. The end result is a very small, fast UPnP stack that provides a specific set of UPnP interfaces.

Reduced Run-Time Size

A further benefit of Intel Device Builder generated code stacks is their significantly reduced run-time size. Techniques such as zero-copy parsing and single-threaded design contribute to a small run-time footprint. The single-threaded design also makes it easier for porting the generated Posix source code to other embedded operating systems, such as VxWorks*, PSOS*, and Nuclius*.

Robust

Despite their single-threaded design, the UPnP stacks generated by Intel Device Builder can handle numerous inbound and outbound HTTP transfers simultaneously, providing a fast response to requests and handling of events.

AV Microstacks

For developers interested in building UPnP AV solutions, having a highly efficient, custom built UPnP core stack is a great start. However, the UPnP AV specification is embodied in several documents and exceeds a couple hundred pages, making it very difficult for inexperienced and untrained developers to quickly deploy interoperable UPnP AV solutions.

The Fast-Track Solution

To provide vendors with a fast-track, Intel has built a set of UPnP AV stacks on top of generated core stacks. Referred to as AV Microstacks, they are very efficient at handling the logic associated with a UPnP AV device or control point. The modules in AV Microstacks contain such tasks as event construction, event moderation, playlist parsing, metadata generation, and metadata parsing. AV Microstacks allow developers to leverage Intel's key learnings and reduce time-to-market for products that need UPnP AV features.

Four Basic Modules

There are four basic modules in AV Microstacks that correspond to the four different roles in UPnP AV:

- Media Renderer Device
- Media Server Device
- Media Renderer Control Point
- Media Server Control Point

Sample Applications—Speedy Retrofit Solutions

To demonstrate these modules in a solution, Intel provides a number of sample applications that build on the AV Microstacks. This allows developers to compile, run, and study the AV Microstacks in action. Generally, these stacks can be integrated into an existing product's solution within a few hours if no porting work is required for the generated source code.

Minimal Code and Run-Time Size

A key feature of the AV Microstacks is minimal code and run-time size. This is a result of a design and review process intent on minimizing binary size and run-time memory use. In line with these objectives, the AV Microstacks are not built with the full feature set offered by UPnP AV. Rather, the solutions implement a reasonable subset of UPnP

AV features.

Customizable Stacks

Intel provides the complete source code for any AV Microstack in the Authoring Tools, allowing developers to customize a stack to fit their needs. Vendors can add features or remove unnecessary code to build the smallest and most efficient solution for their own device. Part of the customization may also involve re-generation of the core portion of the UPnP stack using Intel Device Builder.

Sample Applications

The Intel Authoring Tools for UPnP Technologies offer a number of UPnP solutions. The sample applications target Win32, Posix, and PocketPC. These solutions include the source code to demonstrate how developers can use generated UPnP stacks and AV Microstacks in a complete solution. As well as being convenient starting points for some development projects, the sample applications are also examples of solutions with small memory footprints.

Intel® Micro Light

Intel Micro Light, a home automation device, is often used as a simple learning sample. Full source code is provided for Win32 (80k) and PocketPC (57k). The Win32 version at 80k is a good example of the overhead incurred when adding basic UPnP device interfaces to an existing product.

Intel® Device Scanner

Intel Device Scanner makes use of a modified control point stack to search for and report all of the UPnP devices and services available on the network. It looks very much like the Device Spy tool provided with the Intel Tools for UPnP Technologies, but it is not capable of receiving events or invoking UPnP actions on devices. Full source code is provided for Win32 (180k) and PocketPC (62k).

Intel® Micro Media Server

Other samples use Intel Micro Media Server to allow sharing of local files to the UPnP network. This application can also stream media over HTTP using the internal web server provided with the generated UPnP stack. Running this application allows control points to discover content and the devices available to download the content. Intel Micro Media Server is a starting point for adding UPnP Media Server support to music, picture and video applications. Full source code is provided for Win32 (152k), Posix, and PocketPC (78k).

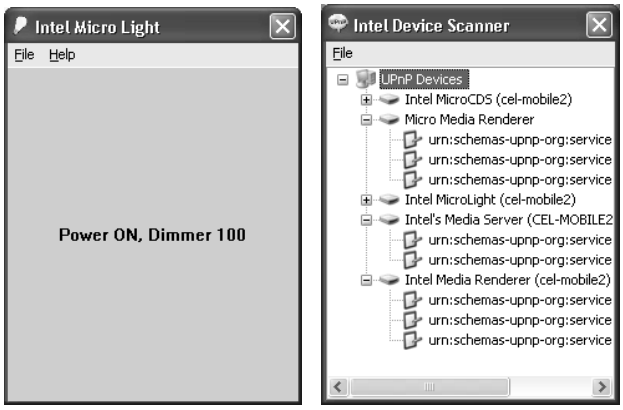

Figure 1. Intel Micro Light on the left; Intel Device Scanner on the right.

Intel® Micro Media Renderer

A UPnP renderer stack is used to advertise the presence of a media sink device on the network. The Intel® Micro Media Renderer wraps the Microsoft Windows Media Player* with the AV Media

Renderer stack for an experience similar to a UPnP-enabled streaming media playback device. Full source code is provided for Win32 (92k), Posix and PocketPC (150k). On PocketPC, the run-time footprint is 2.5MB, which includes the entire streaming and rendering engine provided by Windows Media Player. On Posix the sample application includes a simple media player emulator that simulates the behavior of the media rendering engine without playing media.

Intel® Micro Media Browser

Intel® Micro Media Browser is a sample UPnP AV Media Server control point that allows the user to scan the network for Media Servers and browse the content each one offers. This application would be the starting point of software that discovers media on the network. Full source code is provided for Win32 and PocketPC.

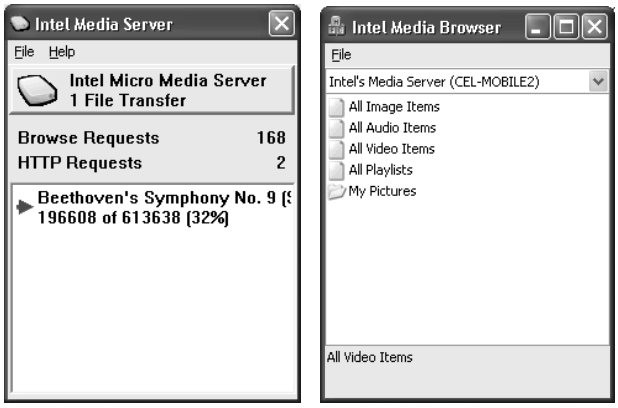

Figure 2. Intel Micro Media Server on the left; Intel Micro Media Browser on the right.

Conclusion

The Intel Authoring Tools for UPnP Technologies are designed to significantly reduce development costs and time-to-market for vendors retrofitting existing products with fully compliant UPnP and UPnP AV functionality. They are easily used by novices and experts alike and are designed to cover all phases of the development and implementation process.

The Intel Authoring Tools for UPnP Technologies are available for download on the Intel web site at: http://www.intel.com/labs/connectivity/upnp/

Appendix

Concepts and Terminology

Control Point (CP): An intelligent network node or application that has the ability to discover and control UPnP devices on the network. UPnP technology makes a clear distinction between a logical device and a Control Point on the network.

Control Point Stacks: Code modules that abstract the specifics of the UPnP protocols for a control point, offering the user a relatively easy API that can be used to discover devices, invoke actions, subscribe to and receive events.

Device: A logical container of services that may contain other devices. All UPnP devices advertise basic information about themselves on the network such as manufacturer and serial number. Some devices also offer a device web page.

Device Stack: A code module that abstracts the specifics of the UPnP protocols for a device. The provided API allows the developer to quickly advertise services, respond to control point invocations, and fire events. Some Device Stacks allow the user to advertise a web page along with the device.

Intel Microstack: Intel embedded UPnP stack, the output of Intel Device Builder.

Services: Logical entities providing a specific service

to the UPnP device network. They are composed of actions that can be invoked on the service and events that a control point can subscribe to.

Service Control Protocol Declaration (SCPD): An XML file that describes a UPnP service. It contains information such as actions, state variables, arguments for each action, and available events. Control Points can download SCPD documents from discovered devices to verify which services, actions, and events are available.

UPnP Device Reference Design: A sample design for a specific type of UPnP device (e.g.: AV media renderer) that can be used by product developers as a basis for their design. Typically a reference design has all the features and capabilities of the UPnP device it references.

UPnP Stack: A generic term used to describe a Device Stack or Control Point Stack, or a combination of both.

Universal Control Point (UCP): A generic or universal type of Control Point that can control all UPnP devices on the network. Universal Control Points are generally used when developing and testing UPnP devices. Intel Device Spy is a good example of a UCP.

Operational Phases of UPnP Devices

The operation of UPnP devices is divided into the following phases:

Discovery: In this phase, UPnP control points search for UPnP devices and services, and UPnP devices advertise their services.

Description: Once a UPnP control point finds a UPnP device or service of interest, it requests its description document. UPnP devices and services respond by sending XML description documents that define the actions and attributes they support.

Control: In this phase, UPnP control points invoke the actions described in the XML description documents associated with the services they control. These actions are executed by the services and typically cause changes in the service states and attributes.

Eventing: UPnP control points subscribe to event servers hosted by the services, which allows them to receive events from a specific service they are interested in. Similar to control actions, events are defined in the corresponding STs (Service Templates).

Presentation: Finally, UPnP devices may choose to host an HTML document that provides a simple GUI for the device.